IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Alexander, III et al.**	§	
	§	Group Art Unit: **2191**
Serial No.: **10/777,909**	§	
	§	Examiner: **Rampuria, Satish**
Filed: **February 12, 2004**	§	
	§	Confirmation No.: **6082**
For: **Method and Apparatus for Removal of Asynchronous Events in Complex Application Performance Analysis**	§	

35525

PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on October 13, 2008.

A fee of $540.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447.

No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

This appeal has no related proceedings or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

The claims in the application are: 1-24

B. STATUS OF ALL THE CLAIMS IN APPLICATION

Claims canceled: 2, 3 and 9-24

Claims withdrawn from consideration but not canceled: None

Claims pending: 1 and 4-8

Claims allowed: None

Claims rejected: 1 and 4-8

Claims objected to: None

C. CLAIMS ON APPEAL

The claims on appeal are: 1 and 4-8

STATUS OF AMENDMENTS

An Amendment to the Final Office Action of July 22, 2008, was not filed. Accordingly, the claims on appeal herein are as presented in the Response to Office Action filed March 12, 2008.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to a method, in a data processing system, for generating a minimized call tree data structure from trace data obtained from a plurality of executions of a computer program (see Step **1910**, **Figure 19**; Specification, page 51, lines 17-20). A plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program is obtained (Step **1920**, **Figure 19**; Specification, page 51, lines 20-23). A minimized call tree data structure is generated from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures (Step **1930**, **Figure 19**; Specification, page 51, lines 23-27). The minimized call tree data structure is outputted (Step **1940**, **Figure 19**; Specification, page 51, line 27-page 52, line 1).

B. CLAIM 4 - DEPENDENT

The subject matter of claim 4, which depends from claim 1, recites that generating the minimized call tree data structure includes copying a first call tree data structure, and walking a second call tree data structure over the first call tree data structure to generate the minimized call tree data structure (Step **1930**, **Figure 19**; Specification, page 51, lines 23-27; see also page 49, line 25-page 50, line 4).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The ground of rejection to review on appeal is as follows:

A. GROUND OF REJECTION 1

The Examiner has finally rejected claims 1 and 4-8 under U.S.C. § 103(a) as being unpatentable over Alexander, III et al., U.S. Patent No. 6,338,159, in view of Kazi et al., "JaViz: A client/server Java profiling tool."

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1 and 4-8)

The Examiner has finally rejected claims 1 and 4-8 under U.S.C. § 103(a) as being unpatentable over Alexander, III et al., U.S. Patent No. 6,338,159 (hereinafter "Alexander"), in view of Kazi et al., "JaViz: A client/server Java profiling tool" (hereinafter "Kazi").

A.1. Claims 1 and 4-8

In finally rejecting the claims, the Examiner states with respect to claim 1:

> Alexander(1) discloses:
> 1. A method, in a data processing system, for generating a minimized call tree data structure from trace data obtained from a plurality of executions of a computer program, comprising:
> obtaining a plurality of call tree data structures (col. 2, lines 32-33 "a call stack...one or more nodes in the tree") corresponding to the trace data (col. 2, lines 28-29 "trace information ...obtained ...") for the plurality of executions of the computer program (col. 2, lines, 38-39 "...number of Java bytecodes executed in each method ... called").
>
> Alexander(1)does not disclose generating a minimized call tree data structure from the plurality of call tree data structures wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures; and outputting the minimized call tree data structure.
>
> However Kazi discloses in an analogous computer system generating a minimized call tree data structure from the plurality of call tree data structures (Kazi page 100 "Tree generation ... merged trace files to create an output file containing the dynamic execution tree") wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures (Kazi page 100 "Tree generation ... Run-time statistics generation...Each detailed ... trace file is analyzed to gather the total number of calls made to each method, the maximum, minimum, and average execution times, and the standard deviation of the execution time for each method"). Thereby minimizing the display for execution.; and outputting the minimized call tree data structure (Kazi page 100 "Tree generation ... merged trace files to create an output file containing the dynamic execution tree").
>
> Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of generating a minimized call tree data structure from the plurality of call tree data structures wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures; and outputting the

minimized call tree data structure as taught by Kazi into the method for providing the trace information as taught by Alexander(1). The modification would be obvious because of one of ordinary skill in the art would be motivated to generate a minimize call tree data structure from the trace data to provide a performance analysis tool to allow developer to determine the execution times have high of low variance as suggested by Kazi (page 115, "Conclusion").

Final Office Action dated July 22, 2008, pages 5-7.

Claim 1 on appeal herein is as follows:

1. A method, in a data processing system, for generating a minimized call tree data structure from trace data obtained from a plurality of executions of a computer program, comprising:

obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program;

generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures; and

outputting the minimized call tree data structure.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). The prior art reference (or references when combined) must teach or suggest all the claim limitations. *In re Royka,* 490 F.2d 981, 180 USPQ 580 (CCPA 1974). In determining obviousness, the scope and content of the prior art are… determined; differences between the prior art and the claims at issue are… ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background the obviousness or non-obviousness of the subject matter is determined. *Graham v. John Deere Co.*, 383 U.S. 1 (1966). "Often, it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art, all in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue." *KSR Int'l. Co. v. Teleflex, Inc.*, No. 04-1350 (U.S. Apr. 30, 2007). "*Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support*

the legal conclusion of obviousness. Id. (citing *In re Kahn*, 441 F.3d 977, 988 (CA Fed. 2006))."

In the present case, the Examiner has not established a *prima facie* case of obviousness in rejecting the claims because neither Alexander nor Kazi nor Alexander in view of Kazi teaches or suggests all the claim limitations. With respect to claim 1, for example, neither Alexander nor Kazi nor Alexander in view of Kazi teaches or suggests "obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program", or "generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures."

In finally rejecting the claims, the Examiner asserts that Alexander discloses "obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program" at col. 2, lines 32-33, col. 2, lines 28-29 and col. 2, lines38-39; and specifically refers to col. 2, lines 38-39 as disclosing "for the plurality of executions of the computer program. Appellants respectfully disagree. Column 2, lines 28-46 of Alexander is reproduced blow for the convenience of the Board:

> The trace information, whether obtained statically or dynamically, is represented as a tree of events. For example, the tree structure of the present invention may reflect the call stacks observed during a program's execution. A call stack is represented by one or more nodes in the tree, and statistics regarding the time spent in the various routines and call stacks is stored at each node.
>
> The present invention may be used to present many types of trace information in a compact manner which supports performance queries. For example, rather than keeping statistics regarding time, tracing may be used to track the number of Java bytecodes executed in each method (i.e. routine) called. The tree structure of the present invention would then contain statistics regarding bytecodes executed. Tracing may also be used to track memory allocation and deallocation. An advantage of this invention is that it is not required to store a trace record for every memory allocation or deallocation, though such an approach is possible under the present invention.

The above recitation may describe that trace information is represented as a tree of events, and that the tree structure "may reflect the call stacks observed during a program's execution" (Emphasis added). However, nowhere in the above recitation does Alexander disclose or suggest obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of

the computer program as recited in claim 1. To the contrary, Alexander is directed to providing trace information in connection with a single execution of a long running program. This is clearly stated, for example, in col. 2, lines 54-61 of Alexander, reproduced below:

> The use of dynamic tracing and reduction, along with dynamic pruning in some cases, is especially usefull in profiling the performance characteristics of long running programs. By using dynamic tracing and reduction (and perhaps dynamic pruning), an accurate and informative performance profile may be obtained for a long running program. (Emphasis added)

Alexander is not directed to generating trace data for a plurality of executions of a computer program and does not disclose or suggest "obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program" as recited in claim 1.

In finally rejecting claim 1, the Examiner appears to suggest that the statement in Alexander that "tracing may be used to track the number of Java bytecodes executed in each method (i.e. routine) called" is a teaching of "the plurality of executions of the computer program" as recited in claim 1. Appellants respectfully disagree. The above recitation refers to plural executions of bytecode not to a plurality of executions of the program itself.

Therefore, Alexander does not teach or suggest "obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program."

Kazi also does not teach or suggest "obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program." Kazi describes a performance analysis tool for generating execution traces, and teaches providing a graphical display of a program execution tree for an entire distributed application (see, for example, the Abstract on page 96 of Kazi). Kazi, however, also does not disclose or suggest "obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program" as recited in claim 1, and does not supply this deficiency in Alexander.

Therefore, neither Alexander nor Kazi nor their combination teaches or suggests "obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program", and claim 1 is not obvious over Alexander in view of Kazi for this reason.

Alexander in view of Kazi also do not teach or suggest "generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures" as recited in claim 1. In finally rejecting the claims, the Examiner acknowledges that Alexander fails to teach or suggest this feature. However, the Examiner believes that Kazi teaches this feature. Appellants respectfully disagree.

The Examiner cites to the following portions of Kazi:

> *Tree generation*. The tree generation step analyzes the merged trace files to create an output file containing the dynamic execution tree for a given client or server program…
>
> *Run-time statistics generation*…Each detailed .prf trace file is analyzed to gather the total number of calls made to each method, the maximum, minimum, and average execution times, and the standard deviation of the execution time for each method.

Kazi, page 100.

The Examiner asserts that the above-cited portions of Kazi teach "generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures," as recited in claim 1. However, Kazi, when viewed as a whole, teaches a method that provides the ability to generate an execution tree that shows all the trace execution threads across Java Virtual Machine (JVM) boundaries, in a large-scale client/server environment using the Java remote method invocation (RMI) facility. Implementation of the method described in Kazi, requires merging three trace files: the detailed trace, the client profile, and the server profile. These files are merged to produce one detailed trace for each JVM. After the merge step is completed, a dynamic execution tree is produced that shows *every* RMI trace execution call recorded on each JVM.

Kazi, however, fails to make up for Alexander's acknowledged deficiencies because Kazi also does not teach or suggest, either in the portions cited by the Examiner or elsewhere in the publication, "generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that

are consistent between the plurality of call tree data structures" as recited in claim 1. Instead, Kazi only teaches the generation of an execution tree that shows *every* trace execution call recorded on each JVM.

In responding to Appellants' arguments that Alexander in view of Kazi fails to teach or suggest "generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures", the Examiner states:

> Kazi teaches "To facilitate the performance analysis of the call graph, statistical information about each of the methods in the program is gathered in the merge step. Each detailed .prf trace file is analyzed to gather the total number of calls made to each method, the maximum, minimum and average execution times, and the standard deviation of the execution time for each method." (see Kazi, page 100, Run-time statistics generation"). Therefore, in regard to the merge step cited, it is clear that Kazi's discloses averaging specific information of a call, graph by totaling the data and computing the average. Averaging the various runs (various values of a respective node) of a computer program inherently minimizes variations (each value relating to a run) in the profile data (trace data).

Final Office Action dated July 22, 2008, page 4.

The portion of Kazi referred to by the Examiner at best teaches that statistics may be gathered and displayed by a visualizer when a node is selected in a call graph. Kazi does not disclose or suggest, either in the portion referred to by the Examiner or elsewhere in the publication, "generating a minimized call tree data structure" as recited in claim 1.

Furthermore, and as discussed previously, Kazi does not disclose obtaining a plurality of call tree data structures corresponding to trace data for a plurality of executions of a computer program; and, therefore, also cannot disclose "generating a minimized call tree data structure from the plurality of call tree data structures.

Therefore, claim 1 is not obvious over Alexander in view of Kazi for this reason also.

For at least all the above reasons, claim 1 is not obvious over Alexander in view of Kazi, and the Examiner has failed to establish a *prima facie* case of obviousness in rejecting the claim. Claim 1, accordingly, patentably distinguishes over the cited references in its present form.

Claims 4-8 depend from and further restrict claim 1 and patentably distinguish over the cited references, at least by virtue of their dependency from claim 1.

A.2. Claim 4

Claim 4 depends from and further restricts claim 1, and is as follows:

> 4. The method of claim 1, wherein generating the minimized call tree data structure includes:
>
> copying a first call tree data structure; and
>
> walking a second call tree data structure over the first call tree data structure to generate the minimized call tree data structure.

In finally rejecting claim 4, the Examiner states:

> **Per claim 4:**
>
> The rejection of claim 1 is incorporated and further, Alexander discloses: 4. The method of claim 1, wherein generating the minimized call tree data structure includes: copying a first call tree data structure; and walking a second call tree data structure over the first call tree data structure to generate the minimized call tree data structure (col. 6, lines 27-29 "the tree is traversed (i.e., walking) to the parent (using the parent pointer), and the current tree node is set equal to the parent node (step 178)... the tree can be dynamically pruned in order to reduce the amount of memory dedicated to its maintenance (step 179).").

Final Office Action dated July 22, 2008, page 7.

Column 6, lines 27-31 of Alexander is as follows:

> If it is an exit event, the tree is traversed to the parent (using the parent pointer), and the current tree node is set equal to the parent node (step **178**). At this point, the tree can be dynamically pruned in order to reduce the amount of memory dedicated to its maintenance (step **179**).

This cited portion of Alexander discloses a method of traversing, i.e. walking, a tree after first determining whether the trace event is an *exit* event. Alexander further teaches that the tree is pruned only where the trace event has been identified as an exit event. However, as described above, no teaching or suggestion is made in Alexander of generating a minimized call tree data structure, and, accordingly, Alexander also cannot teach or suggest "walking a second call tree data structure over the first call tree data structure to generate the minimized call tree data structure" as recited in claim 4. Accordingly, Alexander fails to teach or suggest all the features of claim 4, and the Examiner has failed to state a *prima facie* obviousness rejection of claim 4. Claim 4, therefore, patentably distinguishes over the cited references in its own right as well as by virtue of its dependency.

Claims 5-8 depend from and further restrict claim 4. Claims 5-8, accordingly, also patentably distinguish over the cited art by virtue of their dependency from claim 4.

B. CONCLUSION

For at least all the above reasons, the Examiner has failed to establish a *prima facie* case of obviousness in rejecting claims 1 and 4-8, and it is respectfully requested that the Board reverse the Examiner's Final Rejection of those claims.

Date: December 9, 2008

Respectfully submitted,

/Gerald H. Glanzman/

Gerald H. Glanzman
Reg. No. 25,035
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal is as follows:

1. A method, in a data processing system, for generating a minimized call tree data structure from trace data obtained from a plurality of executions of a computer program, comprising:

obtaining a plurality of call tree data structures corresponding to the trace data for the plurality of executions of the computer program;

generating a minimized call tree data structure from the plurality of call tree data structures, wherein the minimized call tree data structure includes a minimum set of nodes that are consistent between the plurality of call tree data structures; and

outputting the minimized call tree data structure.

4. The method of claim 1, wherein generating the minimized call tree data structure includes:

copying a first call tree data structure; and

walking a second call tree data structure over the first call tree data structure to generate the minimized call tree data structure.

5. The method of claim 4, wherein walking the second call tree data structure over the first call tree data structure includes:

for each node that exists in both the first call tree data structure and the second call tree data structure, generating a node in the minimized call tree data structure and associating values with the node.

6. The method of claim 5, wherein the values associated with the node are values that correspond to the minimum of the values associated with corresponding nodes in the first call tree data structure and the second call tree data structure.

7. The method of claim 4, wherein walking the second call tree data structure over the first call tree data structure includes:

for each node that exists in only one of the first call tree data structure and the second call tree data structure, inhibiting creating a node in the minimum call tree data structures.

8. The method of claim 6, wherein the values associated with each node in the minimized call tree data structure include a minimum base value, a minimum number of calls, a minimum cumulative value, and a minimum absolute cumulative value.

EVIDENCE APPENDIX

This appeal brief presents no additional evidence.

RELATED PROCEEDINGS APPENDIX

This appeal has no related proceedings.